TEKNILLINEN TIEDEKUNTA

Parametrisen suunnittelun hyödyntäminen teräskattoristikon mallinnuksessa

Lauri Kantola

KONETEKNIIKAN TUTKINTO-OHJELMA

Kandidaatintyö

Toukokuu 2021

TIIVISTELMÄ
OPINNÄYTETYÖSTÄ Oulun yliopisto Teknillinen tiedekunta

Koulutusohjelma (kandidaatintyö, diplomityö) Konetekniikan tutkinto-ohjelma		Pääaineopintojen ala (lisensiaatintyö)	
Tekijä Kantola, Lauri		Työn ohjaaja yliopistolla Liedes H., yliopisto-opettaja	
Työn nimi Parametrisen suunnittelun hyödyntäminen teräskattoristikon mallinnuksessa			
Opintosuunta Konetekniikka	Työn laji Kandidaatintyö	Aika Toukokuu 2021	Sivumäärä 40

Tiivistelmä

Työn tarkoituksena on tutkia algoritmiavusteista rakennesuunnittelua. Kirjallisuuskatsauksessa käydään läpi projekteja, joissa parametristä suunnittelua on hyödynnetty, sekä mitä hyviä ja huonoja puolia menetelmän käytöstä on havaittu. Varsinaisessa tutkimuksessa tehdään kattoristikon suunnitteluohjelma käyttäen Rhinoceros 7:n työkalua Grasshopperia. Suunnitteluohjelman toimintaperiaate ja koodi käydään läpi yleisellä tasolla, ja sen toimivuus havainnollistetaan kuvitellulla esimerkkitapauksella. Esimerkkitapaus on hallirakennus, jolle suunnitellaan ja mallinnetaan kattoristikko. Malli viedään SOFiSTiK -laskentaohjelmaan, jossa ristikolle tehdään kevyt rakenneanalyysi. Työssä käydään lyhyesti läpi myös ristikon optimointimahdollisuuksia, minkä parametrinen malli mahdollistaa. Työssä havaittiin parametrisen suunnittelun mahdollistavan nopeamman suunnitteluprosessin tietynlaisissa tapauksissa. Tärkeimpiä työn tuloksia oli huomata, kuinka helppokäyttöinen ja nopeasti opittava visuaalinen ohjelmointikieli voi olla. Toisaalta Grasshopperissa ja siihen lisätyssä SOFiSTiKin lisäosassa on vielä kehitettävää.

Muita tietoia

ABSTRACT
FOR THESIS University of Oulu Faculty of Technology

Degree Programme (Bachelor's Thesis, Master's Thesis) Bachelor's Programme in Mechanical Engineering		Major Subject (Licentiate Thesis)	
Author Kantola, Lauri		Thesis Supervisor Liedes H., University teacher	
Title of Thesis Utilization of parametric design in steel roof truss			
Major Subject Mechanical Engineering	Type of Thesis Bachelor's thesis	Submission Date May 2021	Number of Pages 40

Abstract

The object in this thesis is to find out how parametric design can be utilized in structural modeling and analyzing. In the theory part traditional designing method and parametric designing method are being compared. The thesis showcases some projects where parametric designing have been used and goes through advantages and disadvantages that have been noted in the projects. In the second phase of the thesis steel roof truss designing code is made with Grasshopper, which is a tool working under Rhinoceros 7. The Grasshopper code is explained generally, and the code is tested with case example. The case example is basic hall structure to which steel roof truss is modeled. The model is imported to SOFiSTiK -calculation software in which the truss is analyzed. Optimization opportunities in parametric model are also introduced. In conclusions of the thesis there is found that algorithm aided design may provide faster designing process in projects of certain type. Grasshopper scripting language came out to be easy to learn. However, there is some features that needs more developing.

Additional Information

SISÄLLYSLUETTELO

MERKINNÄT JA LYHENTEET

AAD	*Algorithms-Aided Design,* algoritmiavusteinen suunnittelu. Suunnitteluohjelma, joka hyödyntää valmiita, tai suunnittelijan itsensä luomia algoritmeja.
ahk	Tunnus alapaarteen harjakorkeudelle.
CAD	*Computer-Aided Design,* yleisnimitys tietokoneella käytettäville suunnitteluohjelmille. Kaikki nykyiset suunnitteluohjelmistot ovat CAD-ohjelmistoja.
hk	Tunnus parametrille: *yläpaarteen harjakorkeus*.
jv	Tunnus parametrille: *jänneväli*. Kahden pilarin välinen etäisyys, joiden välille ristikko tulee.
o	Tunnus parametrille: *origo*. Origo on piste, joka määrittää ristikon sijainnin. Tunnuksen perässä voi olla hakasuluissa x, y tai z, esim. o[x], viittaa origon x-koordinaattiin.
SSD	*SOFiSTiK Structural Desktop,* rakenneanalyysiohjelmisto. Tässä työssä käytettiin 2020 versiota.
t	Tunnus parametrille: *tiheys*.
tk	Tunnus parametrille: *tukikorkeus*.

1 JOHDANTO

Nykyään kaikki rakenteiden suunnittelu, mallinnus ja analysointi tehdään digitaalisesti tietokoneilla, käyttäen CAD (*computer-aided design*) -ohjelmistoja. Tämä on helpottanut suunnitteluprosessia huomattavasti verrattuna paperille tehtävään suunnitteluun. Alan uusimpia murroksia on ollut parametrinen suunnittelu, joka mahdollistaa esimerkiksi toistuvien työtehtävien automatisoinnin AAD (*Algorithms-Aided Design*) -ohjelmistoilla, eli algoritmiavusteisilla ohejlmistoilla. Tietokoneella voidaan tutkia eri suunnitteluratkaisut nopeasti ja tehokkaasti, kun rakennemalli on muokattavissa parametrisesti. Tietokone suorittaa suunnittelijan määräämän algoritmin läpi nopeasti ja tarkasti. Perinteinen suunnitteluprosessi on ratkaisulähtöinen, eli suunnitteluprosessin pääasiallinen tarkoitus on tuottaa ratkaisu ongelmaan. Parametrisessä suunnittelussa taas keskitytään itse prosessiin. Prosessia tutkimalla ja muokkaamalla voidaan muuttaa lopputulosta.

Työn tarkoituksena on tutkia parametrista rakennesuunnittelua, ja miten Grasshopper ja SOFiSTiK toimivat yhteen yksinkertaisen kattoristikon mallinnuksessa ja rakenneanalyysissä. Grasshopper on visuaalisen ohjelmoinnin työkalu, joka toimii Rhinoceros 3D nimisen CAD-ohjelman päällä (Rhino 6.0 tai uudempi). Grasshopperissa on esiohjelmoituja komponentteja, joiden syötteitä ja ulostuloja voidaan yhdistellä johdoilla. Visuaalisen ohjelmointikielen etuna on sen helppo ymmärrettävyys ja opittavuus, vaikkei olisikaan aiempaa kokemusta ohjelmoinnista.

Tässä työssä käytetään Rhino 7 versiota, jonka lisäosana toimii Grasshopperin versio 1.0.0007. Parametrisessä rakennesuunnittelussa rakenteen ominaisuuksiin vaikuttaville parametreille annetaan arvoja niin, että lopputuloksena saadaan ominaisuuksiltaan halutunlainen rakenne. Parametrejä voivat olla esimerkiksi rakennuksen korkeus, leveys tai kerrosluku (Lalla 2017). Parametreiksi annetut arvot käyvät läpi määrätyn algoritmin ja antavat siitä riippuvan lopputuloksen. *”Algoritmi on yksityiskohtainen kuvaus tai ohje siitä, miten tehtävä tai prosessi suoritetaan; jota seuraamalla voidaan ratkaista tietty ongelma”* (Wikipedia 2020). Algoritmi antaa aina saman lopputuloksen, ellei parametreja muuteta.

2 KIRJALLISUUSKATSAUS

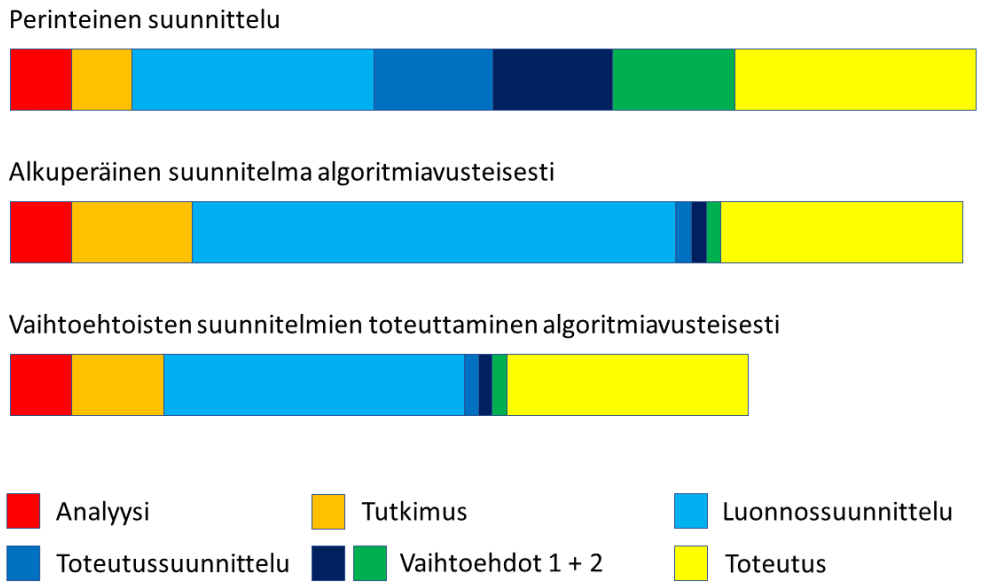
2.1 Parametrinen suunnittelu

Parametrista suunnittelua hyödynnetään usealla eri suunnittelualalla. Esimerkiksi arkkitehdit hyötyvät parametrisesta suunnittelusta, sillä se helpottaa monimutkaisten geometrioiden luomisessa ja eri tilavaihtoehtojen tarkastelussa. Tässä työssä keskitytään kuitenkin siihen, miten rakennesuunnittelija voi hyötyä parametrisesta suunnittelusta.

Alex Lalla (2017) tutki diplomityössään *kantavien rakenteiden parametrinen suunnittelu ja mallintaminen* parametrisen suunnittelun mahdollisuuksia. Tutkimuksen lopputuloksiksi saatiin paljon hyötyjä, joita parametrisella suunnittelulla voidaan saavuttaa. Parametriseen malliin on nopea tehdä muutoksia muuttamalla parametrien arvoja, jolloin algoritmi laskee uuden lopputuloksen reaaliajassa. Parametrinen malli voi olla hyödyllinen, kun projektin edetessä on odotettavissa muutoksia lähtötietoihin tai kaikkia lähtötietoja ei tiedetä. Menetelmällä on myös tehokasta vertailla eri suunnitteluratkaisuja keskenään. Kantavien rakenteiden suunnittelussa parametrisella suunnittelulla viitataan vain siihen, että osa suunnittelijan tehtävistä toteutetaan tai ratkaistaan tietokoneavusteisesti, jolla voidaan nopeuttaa prosessia. Periaatteessa sekä perinteisessä- että parametrisessa suunnittelutavassa täytyy kuitenkin käydä samat asiat lävitse. (Lalla 2017)

Parametrisen suunnittelun suurimmat ongelmat ja heikkouden liittyvät Lallan (2017) mukaan tiedon, taidon sekä kokemuksen puutteeseen kyseisestä asiasta. Suunnittelutoimistoilla ei ole vielä kokemusperäistä tietoa parametriseen suunnitteluun kuluvista työmääristä, joten kuluvan työmäärän ennakointi on hankalaa verrattuna perinteisellä tavalla tehtyyn suunnitteluun. Kuluva työmäärä joudutaan siis arvaamaan heikoin lähtötiedoin, ja huonosti arvioitu työmäärä voi haitata liiketoimintaa. Lisäksi kouluissa ei juurikaan vielä opeteta parametrisiä suunnittelutapoja. Mikäli opiskelijat eivät itse ole opetelleet toimintatapaa, puuttuu heiltä lähtökohtaisesti tietotaito asiasta. (Lalla 2017) En ole havainnut muutosta Lallan (2017) esittämään väitteeseen, ettei koulussa opeteta parametrista suunnittelua.

Parametrisen suunnittelun hyödyntämisen kannattavuus ei ole itsestään selvää. Toteutettaessa malli algoritmiavusteisesti, aikaa kuluu yleensä enemmän ennakkotutkimukseen ja luonnossuunnitteluun verrattuna perinteisesti tehtävään manuaaliseen suunnitteluun. Toisaalta eri vaihtoehtojen läpikäymiseen kuluu vähemmän aikaa parametrisella mallilla, kun suunnitelmaa ei tarvitse työstää aina alusta asti. Suurin etu saavutetaan, kun voidaan tehdä kokonaan uusia suunnitelmia, jotka pohjautuvat samaan prosessiin. Kuvassa 1 vertaillaan perinteistä- ja parametristä suunnitteluprosessia toisiinsa. Kuvasta nähdään prosessien suhteellinen ajankäytön jakaantuminen eri suunnitteluvaiheisiin. (Tanska & Österlund 2014)



Kuva 1. Suunnitteluprosessien ajankäytön vertailu (mukaillen Tanska & Österlund 2014).

Paavo Vähänen (2019) esittää diplomityönsä *Parametrisen suunnittelun hyödyntäminen teräsbetonisten runkorakenteiden luonnossuunnittelussa* johtopäätöksenä, että parametriset suunnittelumenetelmät soveltuvat hyvin rakenteisiin, joissa on paljon toistuvuutta. Myös rakenteen alustava mitoitus ja rakennemallin lähtögeometrian luonti olivat Vähäsen mukaan huomattavasti nopeampaa toteuttaa parametrisesti verrattuna perinteiseen suunnittelumenetelmään. (Vähänen 2019)

Edellä esitettyjen perusteluiden pohjalta voidaan todeta, että valintaan kannattaako projektissa hyödyntää parametristä suunnittelua vai perinteistä suunnitteluprosessia, vaikuttaa useampi tekijä. Parametrisen suunnittelun käyttöä kannattaa harkita, mikäli kaikkia lähtötietoja ei ole vielä selvillä. Näin auki jääneet parametrit voidaan päättää projektin edetessä, tai suunnitelmien muutosten ilmetessä. Kannattaa myös pohtia, voidaanko samaa algoritmia hyödyntää tulevaisuudessa muissakin projekteissa, jolloin pidemmällä aikavälillä voidaan säästää paljonkin suunnitteluaikaa, vaikka lyhyellä aikavälillä algoritmin luomiseen kuluisikin enemmän aikaa. Myös haastavissa ja monimutkaisissa geometrioissa voivat parametriset suunnittelumenetelmät olla käteviä. Mikäli projektissa on suunnittelua, joka sisältää paljon toistoja, puoltaa se parametrisen suunnittelun käytön valintaa. Ristikkorakenteet ovat hyviä esimerkkejä rakenteista, joissa on paljon toistoja.

2.2 Sovellutuksia rakennesuunnittelussa

2.2.1 Kruunuvuorensilta

Kruunuvuorensilta luo kulkuyhteyden Kruunuvuoren asuinalueen, Korkeasaaren, Laajasalon ja Helsingin keskustan välille. Tämän hetken tavoite kulkuyhteyden valmistumiselle on 2026. Sillan pylonin korkeus on 135 m ja sillan pituus on 1.191 km. (WSP 2021)

Mikko Toola (2017) esittää opinnäytetyössään *Teräsrakenteisten siltojen parametrinen mallinnus*, kuinka pylonin mallinnuksessa hyödynnettiin Grasshopperilla tehtyä parametrista mallia. Pylonin teräskotelon teräsosat olivat geometrialtaan haastavia ja ne olivat yksilöllisesti asemoitu koteloon, jonka vuoksi päädyttiin käyttämään parametrisointia. Mallissa pylonin teräskotelo pidettiin lähes kokonaan muokattavissa parametrisesti. Pylonin köydet olivat riippuvuussuhteessa teräskotelon kanssa. Näin teräskotelon siirtäminen vaikutti myös köysien kiinnityskohtaan ja -kulmaan. (Toola 2017)

Toolan (2017) mukaan visuaalisessa skriptauksessa esiintyi myös paljon ongelmia. Projektin Grasshopper-koodi jouduttiin tekemään lähes kokonaan uudestaan, koska koodi ei ollut tarpeeksi selkeästi toteutettu, jolloin sitä oli hankala käyttää ja muokata. Koodin

selkeyteen on kiinnitettävä huomiota, jotta algoritmia voidaan tulkita myöhemmin tai jonkun toisen henkilön toimesta. Tulkintaa helpottaa, kun datavirta pidetään vasemmalta oikealle, vältetään turhia komponentteja ja koodiin lisätään kommentteja, jotka selittävät algoritmia. (Toola 2017)

2.2.2 Olympiastadion

Paavo Vähänen (2019) esittää diplomityössään *Parametrisen suunnittelun hyödyntäminen teräsbetonisten runkorakenteiden luonnossuunnittelussa*, kuinka Helsingin olympiastadionin katsomokatosten teräsrakenteissa hyödynnettiin parametrista suunnittelua. Suunnitteluratkaisuun päädyttiin haastavan geometrian vuoksi. Projektissa myös hyödynnettiin optimointialgoritmeja, joiden avulla kyettiin suunnittelemaan kustannustehokkaita ratkaisuja. (Forsman 2019, Vähäsen 2019 mukaan)

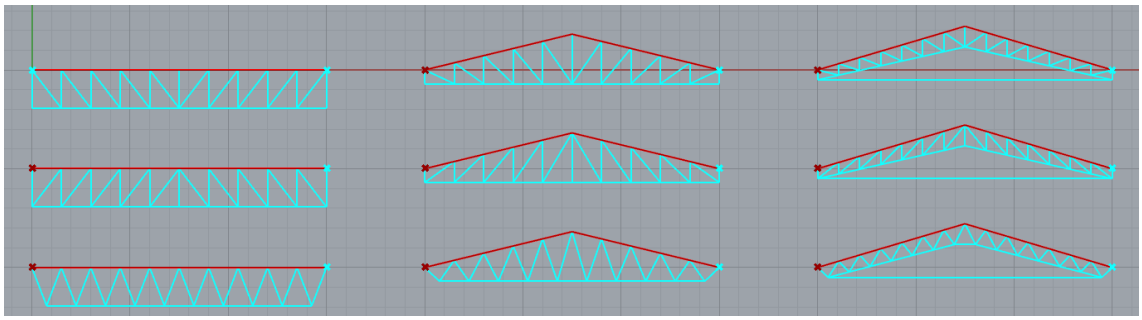
Vähäsen mukaan olympiastadionin katsomokatosten teräsrakenteiden suunnittelun parametrisointi on hyvä esimerkki onnistuneesta parametrisen suunnittelun hyödyntämisestä. Useiden eri ratkaisujen tutkiminen on nopeaa, ja optimointialgoritmeilla voidaan saavuttaa kustannustehokkain rakenneratkaisu. Mitoituksen haasteiksi osoittautui kuitenkin laskentamallin suuri koko ja teräsrakenteiden liitokset olemassa oleviin rakenteisiin. (Forsman 2019, Vähäsen 2019 mukaan)

1 PARAMETRISEN KATTORISTIKON KOODAAMINEN

1.1 Lähtötiedot

Tässä työssä esitettävän Grasshopper-koodin tarkoitus on havainnollistaa parametrisen suunnittelun mahdollisuuksia ristikkorakenteiden suunnittelussa. Ei siis ole tarkoituksen mukaista tehdä koodista kattavaa ristikon suunnitteluohjelmistoa, jolloin työmäärä kasvaisi kohtuuttomaksi. Tehdään muutamia rajauksia, joiden mukainen koodi halutaan luoda. Lisäksi on hyvä huomioida, että työssä rakennettu koodi ei ole ainoa tapa toteuttaa ohjelma. Koodia ei ole optimoitu, eikä arvioitu ulkopuolisen henkilön toimesta, jolloin koodissa voi olla epätehokkaita ratkaisuja.

Koodin tulee kyetä luomaan useampia erilaisia ristikkoratkaisuja. Kuvassa 2 on esitetty ristikkomallit ja -tyypit, joita koodin tulee kyetä mallintamaan. Vasemmalta oikealle kuvassa on palkkiristikot, harjaristikot ja saksiristikot vetotangolla. Ylhäältä alas kuvassa on N-ristikko, RN-ristikko (*Reversed N*) ja K-ristikko. Ristikon diagonaalien ja vertikaalien asemat ristikossa riippuvat siitä, onko kyseessä N-ristikko, RN-ristikko vai K-ristikko. Nämä ovat ainoat ristikkotyypit ja -muodot, joita tässä työssä halutaan tarkastella. Lisäksi ristikon mallinuksessa voidaan tehdä yksinkertaistuksia, esimerkiksi sauvojen kiinnityskohtien osalta.



Kuva 2. Parametrejä muuttamalla saatavat ristikkotyypit.

Tämän työn ristikoissa kaikkien äärimittojen täytyy olla säädettävissä parametrisesti. Näitä mittoja ovat jänneväli, harjakorkeus, alapaarteen harjakorkeus (saksiristikoissa) ja tukikorkeus. Ristikon tiheys, eli kuinka monta diagonaalia ja vertikaalia ristikossa on,

tulee olla säädettävissä. Lisäksi niiden kiinnitys tulee kyetä toteuttamaan joko ideaaliristikon mukaisesti tai jäykästi. Ideaaliristikossa sauvat ovat nivelellisesti kiinnitetty paarteisiin, jolloin niihin ei synny muita vaikuttavia voimia kuin sauvan suuntaisia normaalivoimia. Mikäli diagonaalit ja vertikaalit kiinnitetään ristikkoon jäykästi, niihin syntyy kuormitustilanteessa myös momenttia. Molempia ratkaisuja tulee kyetä vertailemaan koodin avulla.

Ristikon tuentaa halutaan myös tutkia parametrisesti. Ristikon kiinnitys pilareihin tapahtuu ristikon päistä ja kiinnitysten vapausasteet tulee määritellä. Tässä työssä vapausasteiksi valitaan jäykkä-, nivel- ja liukuva nivel. Kiinnitystyypillä on vaikutus ristikon kuormankantokapasiteettiin, sillä se määrittää mitkä voimat ristikon jäykkyyden tulee kyetä kantamaan ja mitä voimia pilareille välittyy. Pilarien tarkastelu jätetään täysin tämän työn ulkopuolelle.

Ristikon poikkileikkauksen profiiliksi valitaan teräksinen suorakaideputki. Suorakaideputken poikkileikkauksen korkeus ja leveys, sekä putken seinämän paksuus tulee olla parametrisesti muutettavissa. Poikkileikkausprofiilin mitoilla on vaikutus putken jäyhyysmomenttiin, joka on oleellinen osa ristikon putkien mitoituksessa. Myös putken teräslaadun on oltava muutettavissa, mutta muut materiaalit kuin rakenneteräkset jätetään tarkastelun ulkopuolelle. Paarteille ja sauvoille tulee pystyä valitsemaan poikkileikkauksen profiilit ja materiaalit erikseen.

Grasshopperiin luotavat kuormat ovat ristikon omapaino, vesikatteen omapaino ja lumikuorma. SOFiSTiK määrittää ja laskee ristikon oman painon materiaalien sekä poikkileikkauksen profiilin mukaan, mutta Grasshopperissa sille luodaan oma kuormitustapaus. Myös kuormitusyhdistelmät tarkastellaan SOFiSTiKin puolella, vaikka ne voitaisiin myös yhdistää Grasshopperissa.

Ristikon optimointi ei ole tässä työssä keskeisessä asemassa, vaan se käydään lävitse vain periaatetasolla. Optimointia ei suoriteta, mutta siihen soveltuvia mahdollisuuksia selostetaan.

Luotu Grasshopper-koodin toimivuus havainnollistetaan esimerkkitapauksella. Esimerkkitapauksena on hallirakenne, johon kattoristikko halutaan suunnitella.

Rakenneanalyysi suoritetaan SOFiSTiKilla. Ristikkoa ei siis mitoiteta, eikä työssä oteta kantaa rakenteen kestävyyteen. Esimerkkitapauksen mukaista hallin kattoristikkoa ei siis tule rakentaa tämän työn pohjalta.

Kaikki tässä työssä käsiteltävät ristikkotyypit ovat bilateraalisesti symmetrisiä, eli peilisymmetrisiä. Sen geometria voidaan luoda määrittelemällä ensin vain toinen puoli, jonka jälkeen geometria voidaan peilata toiselle puolelle. Tämä menetelmä helpottaa ristikon diagonaalisauvojen mallinnusta sekä lyhentää koodin pituutta.

Grasshopper käyttää kolmiulotteista koordinaatistoa, eli jokaisella määriteltävällä pisteellä on oltava x, y ja z koordinaatit. Tässä työssä mallinnettava ristikkorakenne on kaksiulotteinen, joten rakennetta voidaan tarkastella myös tasossa. Valitaan tarkastelutasoksi xz-taso niin, että ristikon jänneväliksi valitaan positiivisen x-akselin suunta ja painovoiman suunnaksi negatiivisen z-akselin suunta. Kaikkien ristikon määrittelypisteiden y-koordinaatti on sama kuin origolla, jonka mukaan muut ristikon pisteet ovat määritelty.

1.2 Parametrit

Tutkimus suoritetaan mallintamalla yksinkertainen parametrinen kattoristikkorakenne Grasshopperilla. Lähtöparametreinä on ristikon tyyppi, mitat, teräslaatu ja profiili. Kaikki parametrit ovat listattuna alla.

1. ristikon tyyppi
    a. N-ristikko (N)
    b. N-ristikko toisinpäin (RN)
    c. K-ristikko (K)
2. origo (o)
3. jänneväli (jv)
4. yläpaarteen harjakorkeus (hk)
5. alapaarteen harjakoekus (ahk)
6. tukikorkeus (tk)
7. tiheys (t)
8. paareet
    a. teräslaatu
    b. neliöputkiprofiilin mitat
9. sauvat
    a. teräslaatu
    b. neliöputkiprofiilin mitat
10. vetotanko
    a. teräslaatu
    b. neliöputkiprofiilin mitat

Näiden edellä esitettyjen parametrien avulla voidaan määritellä kuvan 2 mukaiset ristikkotyypit sekä niiden rakenteelliset mitat ja teräslaadut. Ristikkoa on myös erittäin helppo muokata sen parametrisuuden ansiosta. Tätä voitaisiin hyödyntää erilaisten ristikkotyyppien vertailuissa tai optimoinnissa.

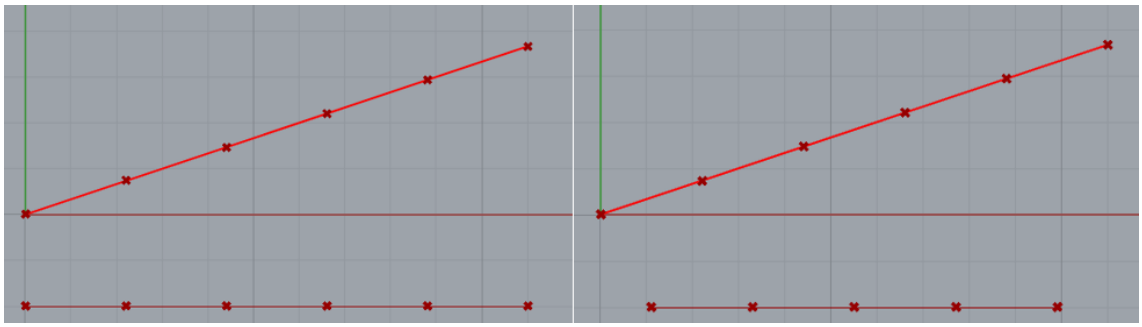
1.3 Ristikon geometrian luonti

1.3.1 Paarteet

Ristikon geometria muodostetaan viivakomponenteilla. Viiva on täysin määritelty, kun sille annetaan aloituspiste ja päätepiste. Geometrian luonti kannattaa aloittaa määrittämällä ensin ylä- ja alapaarteet, sillä näitä voidaan myöhemmin käyttää apuna ristikon diagonaalien ja vertikaalien määrittelyssä. Yläpaarretta mallintavalle viivalle asetetaan aloituspisteeksi origo (o). Yläpaarre kulkee ristikossa räystäältä katon harjalle, eli sen x-koordinaatti on puolet jännevälin (jv) pituudesta, ja z-koordinaatti on yläpaarteen harjakorkeus (hk). Yläpaarretta mallintavan viivan päätepisteen koordinaatti on siis (jv / 2, hk).

Alapaarteen mitat riippuvat ristikkotyypistä sekä tukikorkeudesta (tk) ja alapaarteen harjakorkeudesta (ahk). Ristikkotyypeissä N ja RN alapaarre alkaa samasta x-koordinaatista kuin yläpaarrekkin, mutta K-ristikossa alapaarre alkaa yläpaarteen kahden ensimmäisen sauvojen liitoskohtien puolesta välistä. Alapaarteen aloituspisteen z-koordinaatti riippuu parametristä tk. Alapaarteen aloituspiste on ristikkotyypeissä N ja RN (o[x], -tk), ja K-ristikossa se on (o[x] + (jv / t) / 4, o[z] + tk * (-1)). Merkintä o[x] tarkoittaa origon x-koordinaattia.

Ristikon diagonaalien ja vertikaalien alku- ja päätepisteiden tulee sijaita niin, että paarteita mallintavat viivat kulkevat niiden kautta. Yksinkertaisin tapa varmistaa, että pisteet ovat viivalla, on osittaa viiva *Divide Curve*-komponentilla. Se tarvitsee syötteeksi ositettavan käyrän ja luvun, joka määrää kuinka moneen yhtä suureen osaan käyrä jaetaan. Paarteet halutaan jakaa pätkiksi niin, että jakopisteitä on niissä kohdissa, mihin diagonaali ja/tai vertikaali halutaan kiinnittää. Parametri *tiheys* (t) määrittää, kuinka moneen osaan yläpaarre on jaettu. Esimerkiksi kuvassa 3 näkyy, kuinka tiheyden ollessa t = 5, paarre on jaettu viiteen osaan, eli sauvojen solmupisteitä on yhteensä kuusi kappaletta (t + 1). Kuvassa vasemmalla ovat N- ja RN-ristikkotyyppien paarteet ja oikealla K-ristikkotyypin paarteet.



Kuva 3. Ylä- ja alapaarteen muodostaminen viivakomponenteilla. Viivat ovat jaettu pisteisiin, joihin vertikaalit ja diagonaalit on tarkoitus kiinnittää.

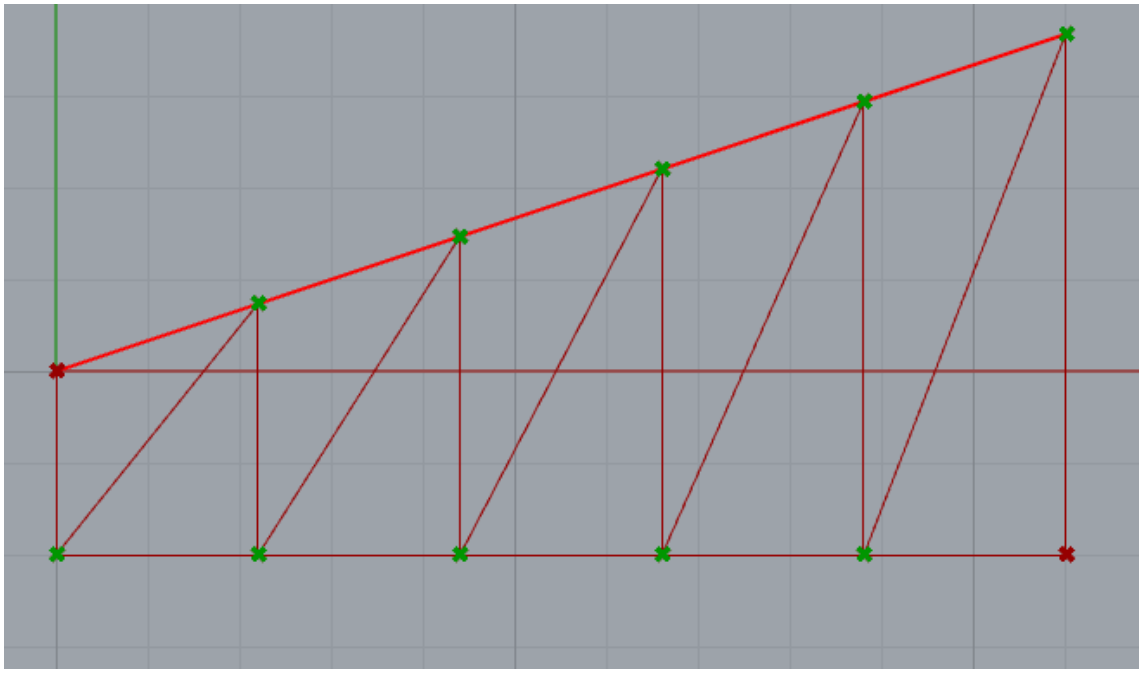
N- ja RN-ristikossa alapaarre halutaan jakaa yhtä moneen solmupisteeseen kuin yläpaarrekkin, mutta koska paarteiden kaltevuuskulma on eri suuri, niin myös paarteiden pituudet ovat eri suuret. Alapaarteen solmupisteitä siirretään niin, että niiden x-koordinaatti on sama kuin yläpaarteen solmupisteillä. Tällöin ristikon vertikaalit saadaan täysin pystysuoriksi. Grasshopperissa voidaan hajottaa piste *Deconstruct*-komponentilla x-, y- ja z-koordinaateiksi, ja vastaavasti koota *Construct Point*-komponentilla piste näistä koordinaateista. Hajottamalla yläpaarteen pisteet koordinaateiksi, voidaan niistä poimia x-koordinaatit, ja syöttää ne *Construct Point*-komponenttiin.

K-ristikossa taas halutaan alapaarteen pisteiden x-koordinaatin olevan yläpaarteen pisteiden puolivälissä. Nyt sauvojen kiinnityspisteitä alapaarteessa on yksi vähemmän kuin yläpaarteessa, ja pisteiden x-koordinaattien siirto tapahtuu samalla tavalla kuin N- ja RN-ristikossa.

1.3.2 Sauvat

Kaikille ristikkotyypeille luodaan oma sauvojen määrittely. N-ristikossa vertikaalit luodaan yksinkertaisesti vain tekemällä viivakomponentti, jonka lähtöpisteinä ovat toisen paarteen solmupisteet ja päätepisteinä toisen paarteen solmupisteet. Sillä, kumpi valitaan lähtöpisteeksi ja kumpi loppupisteeksi, ei ole väliä. Tässä työssä yläpaarteen solmupisteet ovat lähtöpisteinä.

Diagonaalit saadaan muokkaamalla hieman paarteiden solmupisteistä koostuvaa listaa. Kutsutaan tätä listaa pistelistaksi. Tavoitteena on saada sauvaa mallintava viiva alkamaan alapaarteen ensimmäisestä solmusta ja päättymään yläpaarteen toiseen solmuun. *Cull Index*-komponettiä käyttämällä voidaan poistaa listasta alkioita syöttämällä alkion indeksi. Yläpaarteen pistelistasta poistetaan ensimmäinen alkio eli ensimmäinen piste, ja alapaarteen pistelistasta viimeinen alkio eli viimeinen piste. Grasshopperissa ensimmäinen listan alkio saadaan kutsuttua indeksillä 0, ja viimeinen alkio indeksillä -1. Tämän jälkeen pistelistat syötetään *Line*-komponentin alku- ja loppupisteisiin, jolloin saadaan kuvan 4 mukaiset diagonaalit.

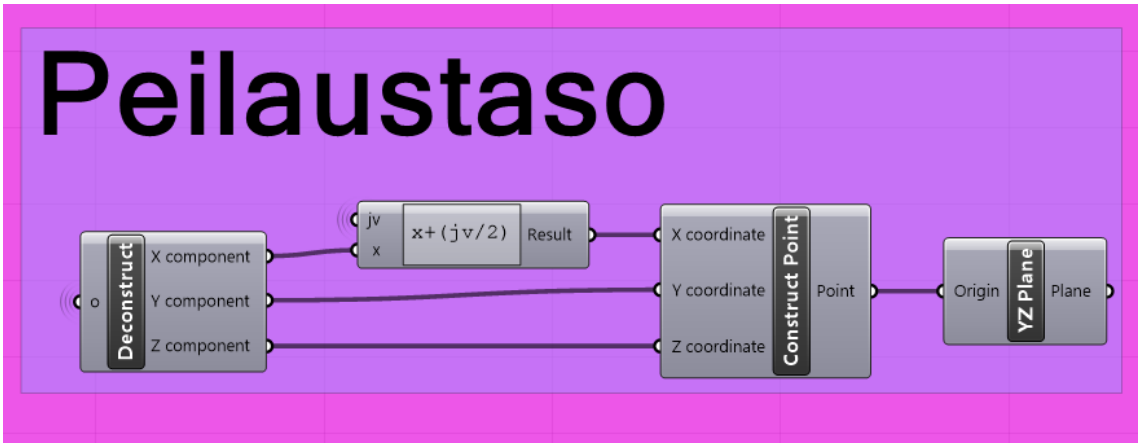


Kuva 4. Diagonaalit ja vertikaalit N-ristikossa.

RN-ristikkotyypin diagonaalit tehdään muuten samalla tavalla kuin N-ristikossa, erona vain se, että yläpaarteen pistelistasta poistetaan viimeinen alkio ja alapaarteen pistelistasta ensimmäinen alkio. Vertikaalit ovat N- ja RN-ristikkotyypeillä samat, joten jo määriteltyjä vertikaaleja voidaan käyttää myös RN-ristikossa. K-ristikossa on vain diagonaaleja, sillä KT-ristikkotyyppi jätetään työn ulkopuolelle. K-ristikon diagonaalit saadaan samalla periaatteella kuin N- ja RN-ristikoiden diagonaalit.

1.3.3 Geometrian peilaaminen

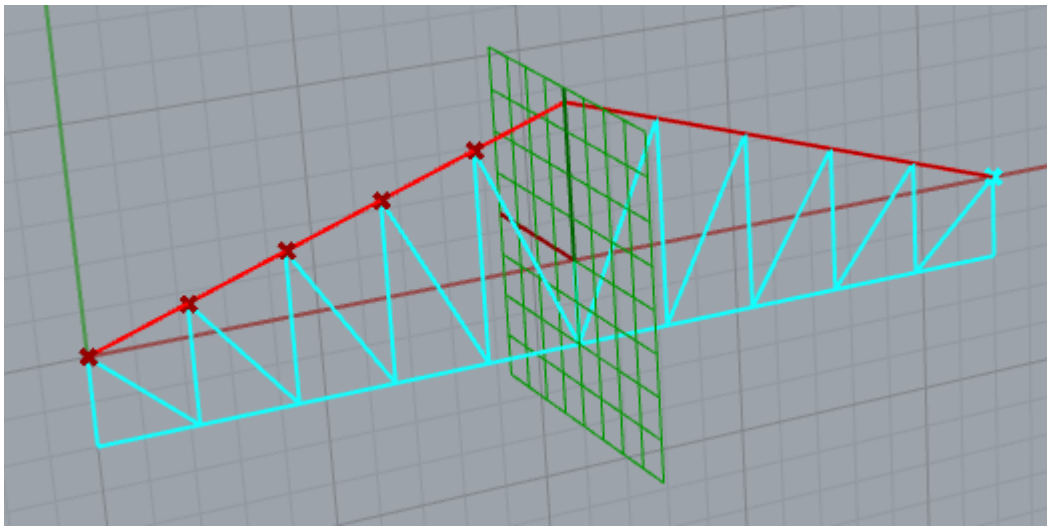
Peilaustason tulee olla yz-tasossa, ja tason yksi piste tulee sijaita ristikon jännevälin puolessa välissä, mihin asti ristikon puolikas on mallinnettu. Grasshopperissa on valmiiksi yz-taso-komponentti, johon tulee syöttää ainoastaan tason origo. Tason origon koordinaatiksi asetetaan piste (jv / 2, 0, 0). (ks. kuva 5).



Kuva 5. Peilaustason määrittely Grasshopperissa.

Geometria peilataan *Mirror*-komponentilla, jonka syötteeksi asetetaan yllä määritetty peilaustaso ja peilattava geometria. Koska ristikkotyyppejä on määritetty nyt useampia, peilataan yksittäiset ristikon osat erikseen. Näin tehtäessä osien kutsuminen koodissa on helpompaa, kun geometria ei ole yhdistetty yhdeksi.

Ennen kuin geometria syötetään peilauskomponenttiin, täytyy peilattavasta geometriasta poistaa peilaustasossa olevat elementit, kuten N- ja RN-ristikoissa oleva keskimmäinen vertikaali, jottei geometriassa ole päällekkäisiä elementtejä. Jos mallissa on päällekkäisiä viivoja, voi se haitata mallin analysointia. Kuvassa 6 on esitetty koodin luoma peilaustaso Rhinossa.



Kuva 6. Peilaustason (vihreällä) paikka Rhinossa. Peilaustason vasemmalla puolella on määritetty geometria ja oikealla puolella geometria peilattuna.

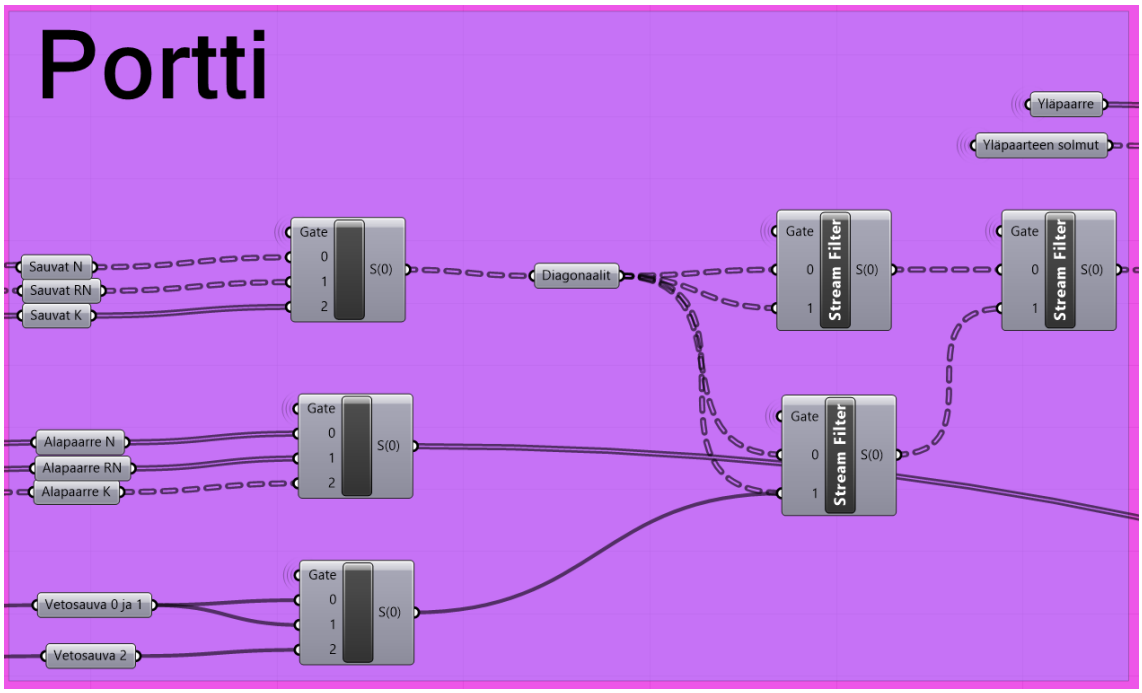
1.3.4 Vetosauva

Vetosauva voidaan määritellä saksiristikon N- ja RN-ristikkotyyppeihin samalla koodilla, mutta K-ristikossa alapaarre on erilainen, ja siihen tarvitaan oma määrittely. Vetotanko on tehokkain, kun sen kiinnityskohta on mahdollisimman lähellä pilareita, mutta alempana olevat vetotangot vähentävät hallin hyötykorkeutta. Käytetään vetotangon kiinnityspisteinä alapaarteen sauvojen kiinnityspisteitä, eli vetosauvaa ei peilata, vaan se luodaan suoraan paikoilleen. Alapaarteen määritetyt pisteet ja peilatut pisteet ovat omissa pistelistoissaan, ja näistä pisteistä voidaan valita, minkä pisteiden välille vetosauva tehdään. Käytetään kahta *List Item*-komponenttia, jonka syötteiksi laitetaan pistelistat, jolloin haluttu piste saadaan syöttämällä kyseisen pisteen indeksi. Indeksin syötteeksi voidaan kytkeä *Number Slider*-komponentti, jolla vetotangon paikka voidaan helposti siirtää.

1.4 Rakennemallin määrittely

Nyt kaikkien ristikkotyyppien kaikki osat ovat määritelty, ja ohjelma suorittaa kaikki määritetyt geometriat. Ohjelmalla halutaan saada aina parametrien mukainen ristikkotyyppi, ja kaikkien muiden ristikkotyyppien geometriat halutaan sivuuttaa

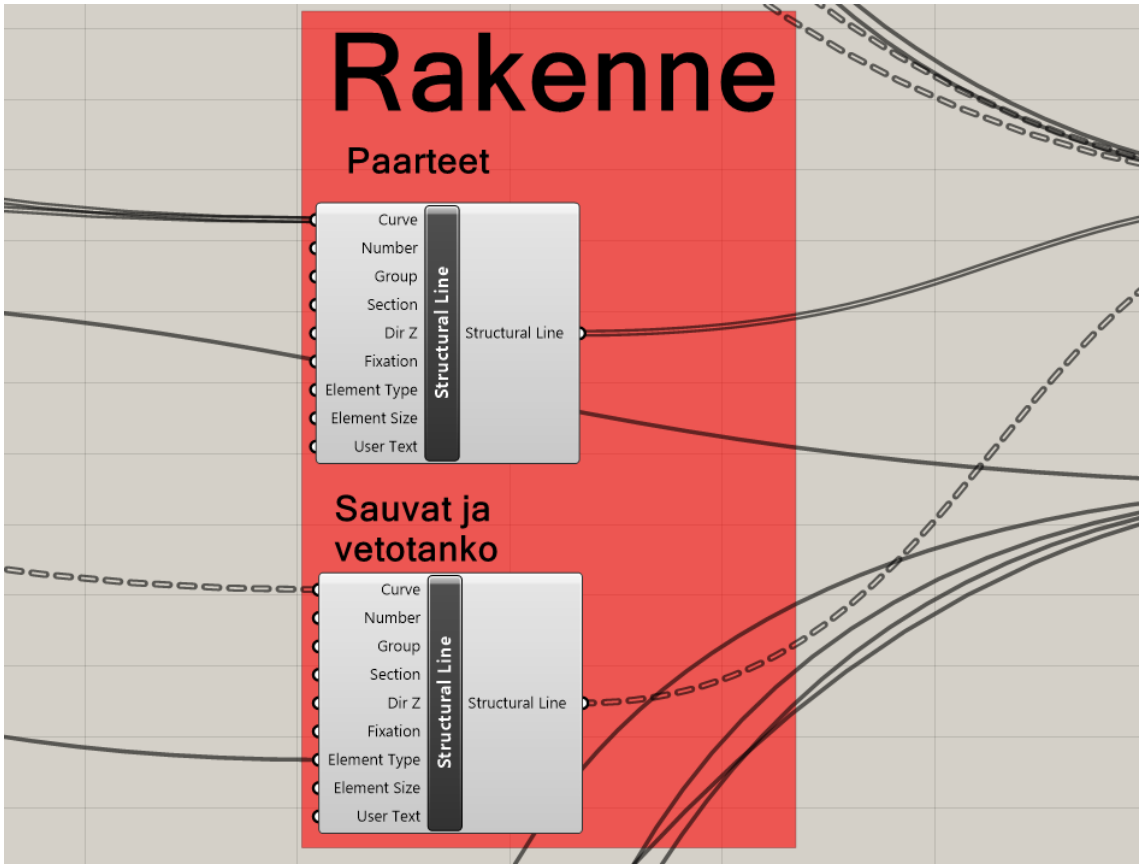
mallista. Tämä voidaan tehdä kuvassa 7 näkyvillä *Stream Gate*-komponenteilla, jotka tarvitsevat syötteiksi portin, joka valitsee, mikä datavirta päästetään lävitse ja minkä läpipääsy estetään. Tässä tapauksessa on kolme ristikkotyyppiä, joten portti tarvitsee kolme numerovaihtoehtoa (N=0, RN=1 ja K=2). Nyt *Stream Filterin* tuloporttiin *0* syötetään kaikki N-ristikkotyypille määritellyt geometriat, tuloporttiin *1* kaikki RN-ristikon geometriat ja tuloporttiin *2* kaikki K-ristikon geometriat.



Kuva 7. Portti määrittää, mistä geometriasta luodaan rakennemalli.

Tähän mennessä geometria on määritetty vain viivakomponenteista, joissa ei ole muuta informaatiota kuin itse viiva. Viivat täytyy määritellä rakenteeksi, jossa on tiedot mm. kyseisen rakenteen poikkileikkauksesta, elementtityypistä ja elementtikoosta. SOFiSTiK lisäosan mukana tulee *Structural Line*-komponentti (kuva 8), jolla on helppo antaa tarvittavia arvoja geometrialle, jotta siitä saataisiin toimiva rakenne. Kaikki ristikon osat, joille halutaan samat rakenteelliset ominaisuudet, voidaan viedä samaan *Structural Line*-komponenttiin. Tässä kohtaa on tärkeä huomioida elementtityyppi. Oletuksena elementtityypiksi on valittu palkkielementti. Tällä on vaikutusta rakenneviivan tuentaan. Paarteiden tulisi olla solmukohdistaan täysin jäykkiä, jonka vuoksi palkkielementti (*Beam*) on hyvä valinta. Ristikon vertikaali- ja diagonaalisauvat ovat ideaaliristikossa

niveltuettuja, jolloin niihin ei muodostu momenttia ollenkaan. Jos elementtityypiksi valitaan ristikkoelementti (*Truss)*, saadaan sauvat käyttäytymään ideaalisesti, jolloin niihin vaikuttaa vain normaalivoimia. Tätä parametria muuttamalla on helppo tarkastella eri elementtityypin vaikutusta ristikkoon.



Kuva 8. *Structural Line*-kompnentit.

1.5 Ristikon tuenta

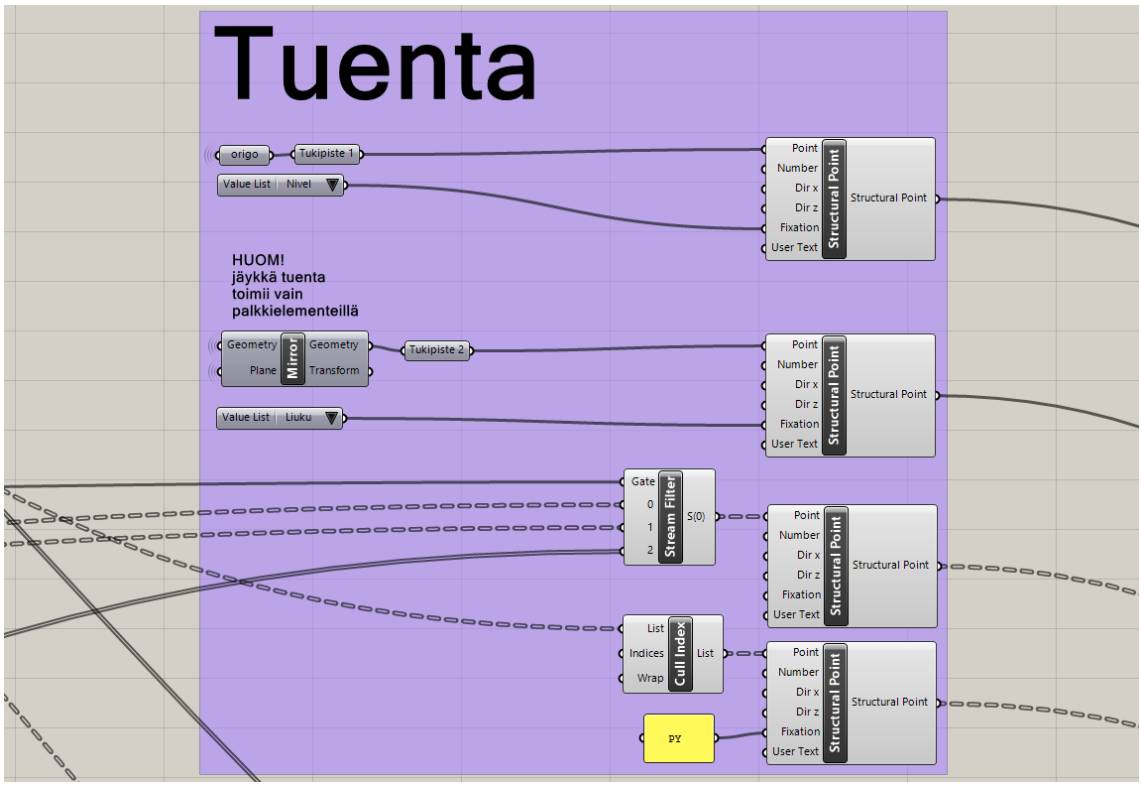
Pilari ja kattoristikko voivat olla liitettynä toisiinsa monella eri tavalla. Liitostyyppi määrittää sen, välittyykö momentti pilarin ja ristikon välillä. Todellisissa rakenteissa osa ristikon momentista välittyy yleensä pilarille. Tämä voitaisiin mallintaa jousituennalla. Yksinkertaistamisen vuoksi tässä työssä tarkastellaan ainoastaan taulukon 1 mukaisia ideaalisia liitostyyppejä. Koska tässä työssä tarkastellaan ristikkoa ainoastaan tasossa, eikä y-suuntaisia voimia vaikuta rakenteeseen, ei tätä vapausastetta tarvitsisi tukea. Todellisuudessa ristikot olisivat tuettu orsilla toisiinsa yläpaarteistaan ja ristikoissa olisi

jäykisteet, jotka estäisivät ristikkojen kaatumisen. Mahdollisen koodin jatkojalostamisen vuoksi tässä työssä on nyt tuettu myös kaikki liike y-suuntaan. Lisäksi on hyvä huomioida, että *Truss* -elementtityypin tuenta ei ole jäykkä, vaikka se niin asetettaisiin. Elementtityyppi on siis hallitseva, mikäli elementtityypin ja tuennan välillä on ristiriita (kuva 9).

Taulukko 1. Tuennan tyypit, vapausasteet ja CADiNP syntaksi.

Tuennan tyyppi	**Tuetut vapausasteet**	**CADiNP syntaksi**
Täysin jäykkä	x, y, z, Rx, Ry, Rz	PPMM
Nivel	x, y, z, Rx, Rz	PPMXMZ
Liukuva nivel	y, z, Rx, Rz	PYPZMXMZ

Tuennan tiedot syötetään *Structural Point*-komponenttiin, joka muokkaa syötetyt arvot CADiNP syntaksin mukaiseen muotoon. *Structural Point*-komponentti tarvitsee tuettavat pisteet ja tuetut vapausasteet CADiNP muodossa. Jotta eri tuentavaihtoehtoja olisi helppo vaihdella ilman CADiNP-koodin kirjoittamista, tallennetaan tuennan koodi valmiiksi *Value List*-komponenttiin (kuva 9).



Kuva 9. Ristikon tuennan määrittely.

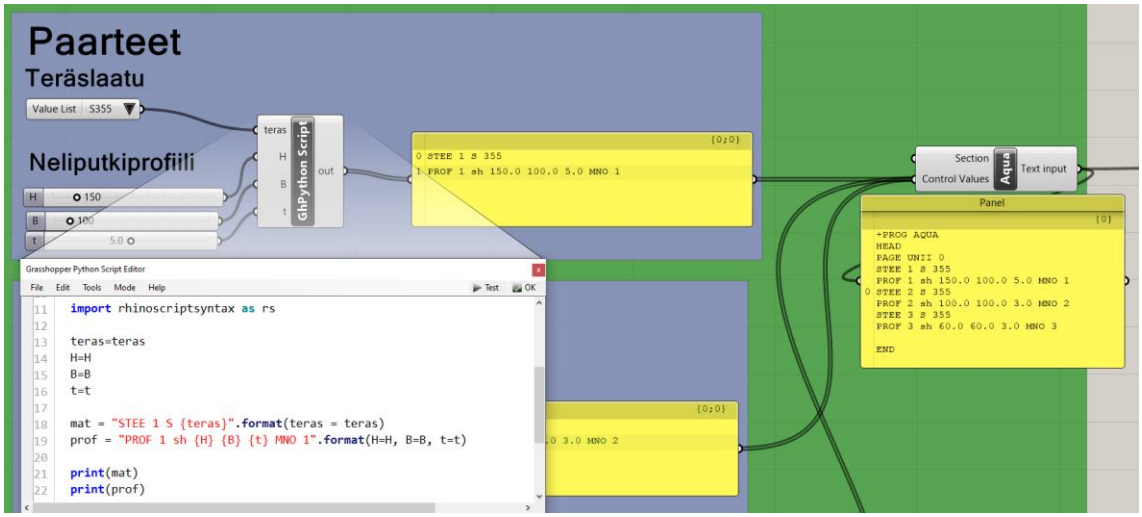
1.6 Materiaalin ja profiilin määrittely

Grasshopperiin asennettava SOFiSTiK lisäosa sisältää poikkileikkauksen määrittämiseen tarkoitettuja komponentteja. Tässä työssä niitä ei kuitenkaan käytetä, sillä tarkastelu rajataan ainoastaan neliöputkiprofiileista tehtyihin poikkileikkauksiin, vaikka mallin parametrisuus antaisikin mahdollisuuden käsitellä helposti myös muitakin profiileja.

SOFiSTiK käyttää tekstisyötössään CADiNP kielistä koodausta. Käyttämällä *GhPython Script*-komponenttia syötettävät poikkileikkausarvot ja materiaalit voidaan muokata CADiNP syntaksin mukaiseksi materiaalin ja profiilin määrittelyksi. Parametreiksi valitaan neliöputkiprofiilin korkeus H, leveys B ja seinämän paksuus T, sekä teräslaatu. Kuvassa 10 on esitetty Grasshopper komponentit, joilla poikkileikkausarvot määritellään paarteille. Paneelissa näkyy CADiNP syntaksin mukainen koodi, joka tarvitaan materiaalitietojen siirtämiseen SOFiSTiKiin. Käytännössä kyseiseen koodiin on tallennettu seuraava informaatio:

- Materiaali nro 1: rakenneteräs S355
- Profiili nro 1: neliöputki 150 mm x 100 mm x 5 mm, materiaali nro 1

Kuvan 10 mukainen Grasshopper-koodi kopioidaan kahdesti, jotta myös sauvoille ja vetotangolle saadaan määritettyä poikkileikkausarvot erikseen. *Aqua*-komponentti luo CADiNP-koodin sille syötetyistä poikkileikkausarvoista, joka voidaan suoraan syöttää *SOFiSTiK Project*-komponenttiin.



Kuva 10. Materiaali- ja poikkileikkausparametrien muunto CADiNP syntaksin mukaiseksi määrittelyksi.

1.7 Kuormat

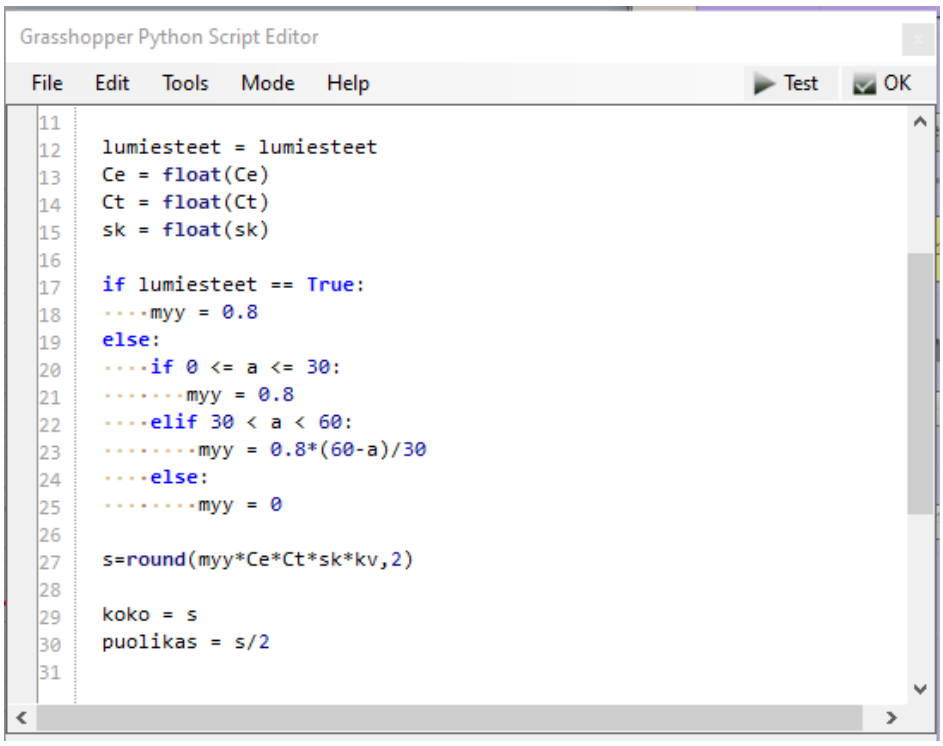
1.7.1 Lumikuorma

Jotta lumikuorman suuruus voidaan määritellä Grasshopperissa, täytyy kuorman suuruuteen vaikuttavat valinnat tuntea. Seuraavaksi käydään läpi yksinkertainen lumikuorman määritys harjakatolle. Eurokoodin SFS-EN 1991-1-3 mukaan kattojen lumikuorma määritetään normaalisti vallitsevissa tilanteissa seuraavasti:

$$s = \mu_i C_e C_t s_k \quad (1)$$

missä μ_i on lumikuorman muotokerroin. Harjakattojen yhteydessä käytetään μ_2
s_k on maanpinnan lumikuorman ominaisarvo
C_e on tuulensuojakerroin
C_t on lämpökerroin

Lumikuorman muotokerroin μ_2 riippuu katon kaltevuuskulmasta. Jos lumen liukuminen pois katolta on estetty, niin lumikuorman muotokertoimelle käytetään vähintään arvoa 0.8. (SFS-EN 1991-1-3 + AC + A1 2015). Lumikuorman muotokertoimen valintaan tarvitaan jos-funktio (*if statement*). Jos katon kaltevuuskulma on $0° \leq \alpha \leq 30°$, valitaan taulukosta lumikuorman muotokertoimen μ_2 arvoksi 0.8 jne. Grasshopperissa voidaan kirjoittaa omaa python koodia *GhPython Scritp*-komponentilla. Tähän komponenttiin voidaan lisätä ja nimetä syötteitä sekä ulostuloja, ja näitä voidaan käyttää koodissa. Kuvassa 11 on esitetty Pythonilla koodattu taulukon 2 mukaiset lumikuorman muotokertoimet.



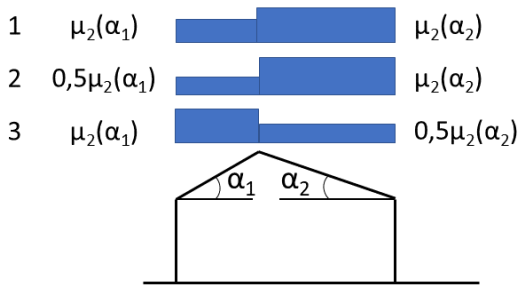
Kuva 11. Grasshopper *Python Script Editor*: lumikuorman määrittely.

Taulukko 2. Lumikuorman muotokertoimet (mukaillen SFS-EN 1991-1-3 + AC + A1 2015).

Katon kaltevuuskulma α	$\mathbf{0° \leq \alpha \leq 30°}$	$\mathbf{30° < \alpha < 60°}$	$\mathbf{\alpha \geq 60°}$
$\mu_1(\alpha)$	$\mu_1(0°) \geq 0{,}8$	$\mu_1(0°)\frac{(60° - \alpha)}{30°}$	0,0
$\mu_2(\alpha)$	0,8	$0{,}8\frac{(60° - \alpha)}{30°}$	0,0
$\mu_3(\alpha)$	$0{,}8 + 0{,}8\frac{\alpha}{30}°$	1.6	--

Harjakattojen lumikuorman laskennassa on tarkastettava kolme kuormitustapausta, jotka ovat esitettynä kuvassa 12. Tässä työssä käsitellään ainoastaan sellaisia harjakattoja, joiden katon lappeet ovat samanlaiset molemmin puolin. Kun katon kulmat $\alpha_1 = \alpha_2$, kuormitustapauksia on kaksi. Ensimmäisessä kuormitustapauksessa tarkastellaan tilannetta, jossa lumi on jakaantunut tasan katon molemmille lappeille, ja toisessa

kuormitustapauksessa tarkastellaan tilannetta, jossa lumi on kinostunut toiselle lappeelle. (SFS-EN 1991-1-3 + AC + A1 2015)



Selite
1 Tapaus (i)
2 Tapaus (ii)
3 Tapaus (iii)

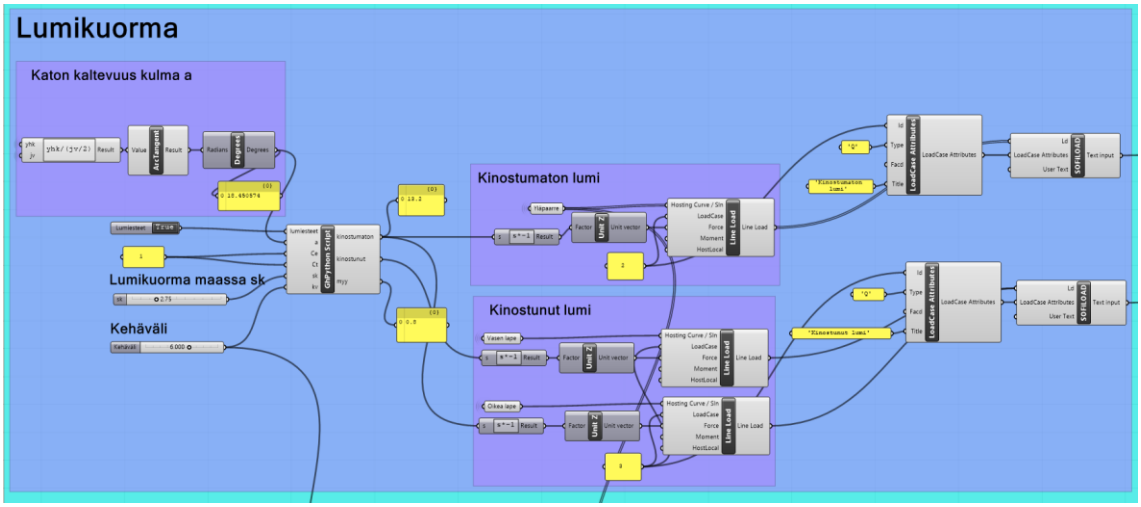
Kuva 12. Harjakaton lumikuorman muotokertoimet (mukaillen SFS-EN 1991-1-3 + AC + A1 2015).

Tässä työssä tarkastellaan ainoastaan yhden ristikon kantamaa kuormaa ja Grasshopperissa mallinnettiin vain yksi ristikkokehä ilman muita kattorakenteita. Lumen aiheuttama kuormitus on helpoin määrittää ristikolle viivakuormituksena. Yksi ristikkokehä kantaa lumikuorman koko jännevälin matkalta ja ristikon molemmilta puolilta puolet kehävälin matkalta.

Lumikuorman laskennan parametrit ovat siis

1. lumiesteet (True/False)
2. maanpinnan lumikuorman ominaisarvo s_k
3. maastotyyppien yhteydessä käytettävä kerroin C_e
4. lämpötilakerroin C_t
5. kehäväli (kv)

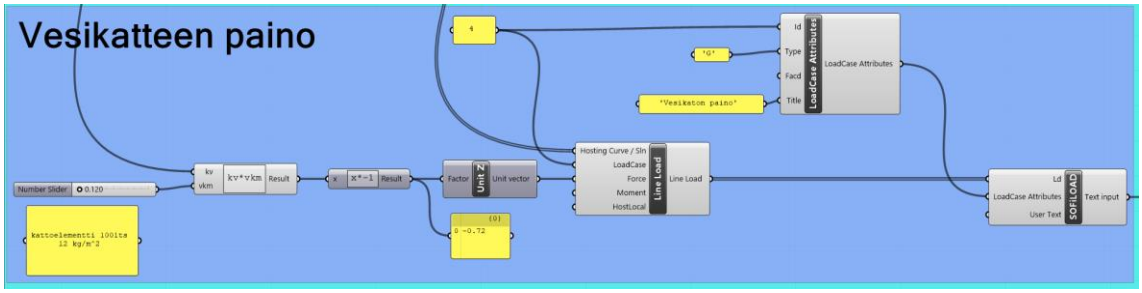
GhPython Script-komponentin ulostuloksi saadaan arvot ”kinostumaton” ja ”kinostunut”. Nämä arvot muunnetaan ensin vektoreiksi, joiden suunta on negatiivisen z-akselin suuntainen. Vektorit syötetään *Line Load*-komponentin syötteeksi *Force*. Tähän lisätään vielä *LoadCase Attributes*-komponentti, johon voidaan määritellä kuormitustapauksen tietoja, kuten kuormitustapauksen numero, tyyppi ja otsikko. Kinostumattomalle lumelle kuormitustapauksen numero on 2 ja kinostuneelle 3. Kuormituksen tyyppi ”Q” viittaa muuttuvaan kuormaan. Tällä on vaikutusta siihen, miten SOFiSTiK määrittää kuormalle varmuuskertoimet kuormitusyhdistelmiin sekä, millaisia rajatiloja kuormitusyhdistelmissä tarkastellaan. Edellä määritetyt tiedot viedään vielä *SOFiLOAD*-komponenttiin, joka muuttaa tiedot SOFiSTiKille ymmärrettäväksi, CADiNP -syntaksin mukaiseksi (kuva 13).



Kuva 13. Lumikuorman määrittely Grasshopperissa.

1.7.2 Vesikatteen oman painon aiheuttama kuorma

Vesikatetta ei mallinneta koodilla mitenkään, vaan sen ainoa parametri on suoraan vesikatteen omapaino neliömetriä kohden [kg/m^2]. Tämä parametri muutetaan viivakuormaksi ristikolle kertomalla sitä parametrillä *kehäväli* (kv), jolloin sen yksiköksi saadaan kilogrammaa metriä kohden [kg/m]. Tämä arvo muunnetaan miinus z-suuntaan osoittavaksi vektoriksi, joka syötetään edelleen *Line Load*-komponenttiin. Koodiin lisätään *LoadCase Attributes*-komponentti, johon syötetään kuormituksen numero, tyyppi ja otsikko. Kuormitustapauksen numero on 4, kuormitus tyyppi on ”G”, joka viittaa pysyvään kuormitukseen, ja kuormitustapauksen otsikko on ”Vesikaton paino”. Lopuksi vielä edellä määritetty informaatio syötetään *SOFiLOAD*-komponenttiin (kuva 14).



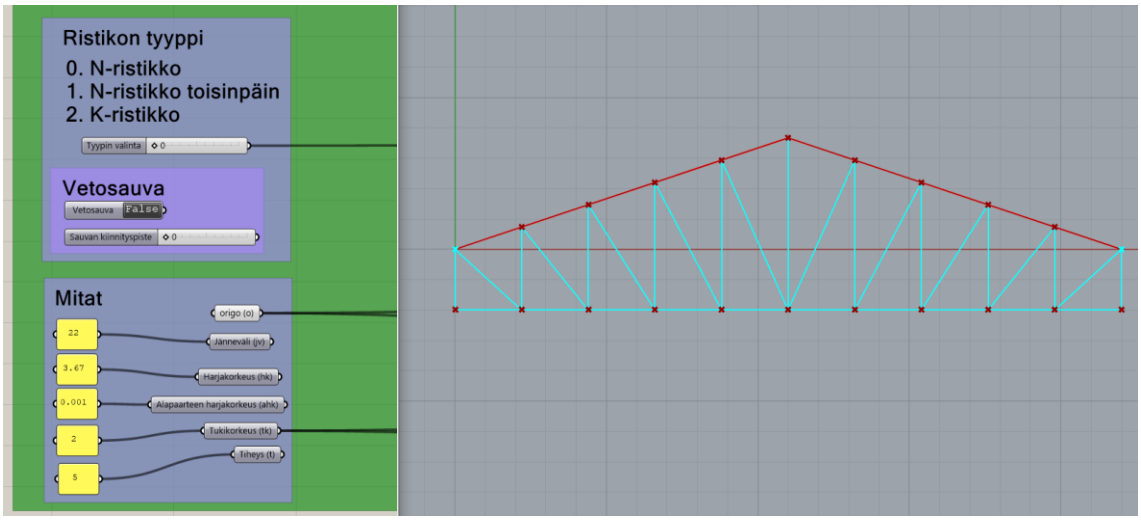
Kuva 14. Vesikatteen omapaino määritelty Grasshopperissa.

2 KOODIN TESTAUS ESIMERKKITAPAUKSEN AVULLA

2.1 Lähtötiedot ja parametrien valinta

Valitaan esimerkin rakenteelle mitat ja sijainti. Rakennuksen sijainnilla on eurokoodin mukaan vaikutus lumikuorman mitoitusarvoon. Tässä työssä ei käsitellä kattorakenteen jäykisteitä, joten sivuttaiset kuormat, kuten tuulikuorma, jätetään tarkastelun ulkopuolelle. Näin ollen esimerkkirakennuksen maastoluokalla ei ole väliä. Ristikon jänneväli on sama kuin hallin leveys, ja hallin pituudella ei ole merkitystä yhtä ristikkoa tarkasteltaessa. Koska tuulikuorma jätetään tarkastelematta, pilareiden korkeudella ei ole myöskään merkitystä. Hallin kehäväli vaikuttaa kuitenkin siihen, kuinka suurelta pinta-alalta lumikuorma vaikuttaa yhdelle ristikolle.

Otetaan esimerkkirakenteeksi halli, joka sijaitsee Oulussa. Valitaan hallin leveydeksi 22 m ja pituudeksi 48 m. Kehien lukumääräksi valitaan 8 kpl, joten kehäväli on 6 m. Katon kaltevuudeksi valitaan 1:3 ja näin ollen katon kaltevuuskulma on 18.4°. Harjakorkeudeksi saadaan jännevälin ja katon kaltevuuden avulla 3.67 m. Edellä esitetyt tiedot syötetään parametreiksi kuvan 15 mukaan. Alkuarvauksena ristikkotyypiksi on valittu N-ristikko sekä parametrille *tiheys* (t) on valittu arvo 5. Parametrille *Alapaarteen harjakorkeus* (ahk) on annettu arvo 0.001, koska laskennassa syntyi epästabiili tilanne arvolla 0. Tutkimuksessa ei selvinnyt, mistä kyseinen tilanne johtui, joten arvoksi valittiin pieni numero.



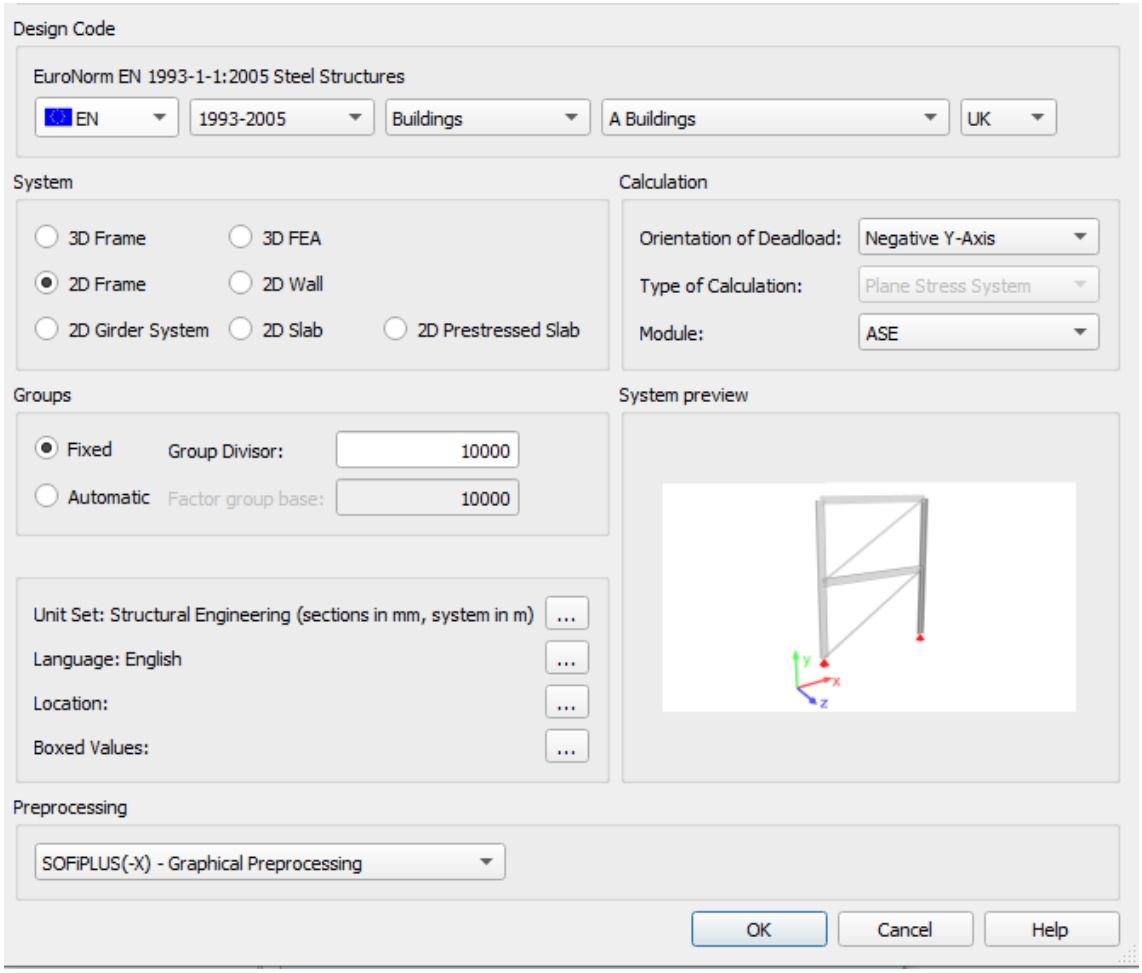
Kuva 15. Esimerkkitapauksen parametrit syötettynä Grasshopper-koodiin, sekä parametrien mukainen ristikko Rhinossa.

Eurokoodin SFS-EN 1991-1-3 standardin kansallisessa liitteessä on esitetty lumikuorman ominaisarvoksi maassa Oulun alueella sk = 2.75 kN/m^2 (Ympäristöministeriö 2019). Tässä työssä tuulikuorman vaikutusta ei tarkastella ollenkaan, eikä esimerkki rakennuksen maastoluokkaa ole määritelty, jonka vuoksi käytetään Eurokoodin taulukon 5.1 antamaa ”normaali maasto” arvoa C_e = 1. Myös lämpökerroin C_t = 1 syötetään lumikuorman parametrit Grasshopperiin.

Vesikate tehdään elementeistä, joiden paino on 12 kg/m^2, joten se aiheuttaa kuormituksen 0.12 kN/m^2. Syötetään saatu kuormitus myös Grasshopperiin.

2.2 SOFiSTiK

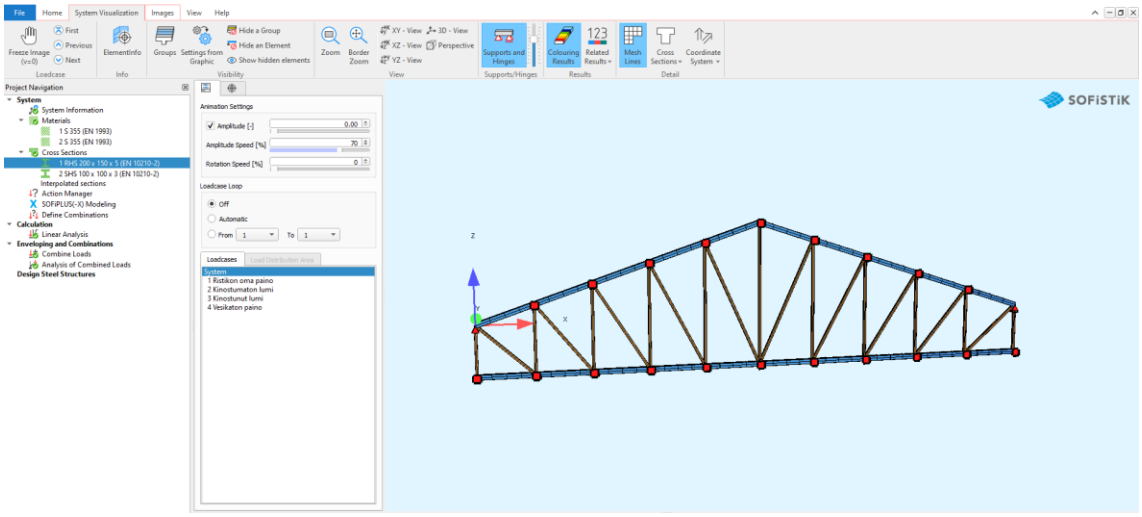
Luodaan SOFiSTiK Structural Desktop (SSD) -ohjelmalla projekti, ja valitaan kuva 16 asetukset. Tässä kohtaa projektiin voidaan valita mm. käytettävät suunnittelukoodit, käytettävät yksiköt ja koordinaattiakselit.



Kuva 16. SSD projektin luonti.

Grasshopper malli saadaan kytkettyä SSD:hen *SOFiSTiK Project*-komponentin avulla, syöttämällä siihen projektin tiedostosijainti. Kun komponenttiin on syötetty oikea tiedostosijainti ja Grasshopper malli lasketaan, saadaan malli siirrettyä SSD:hen, jossa

sille voidaan tehdä erilaisia analyyseja. Kuvassa 17 näkyy malliristikko SSD:n käyttöliittymässä.



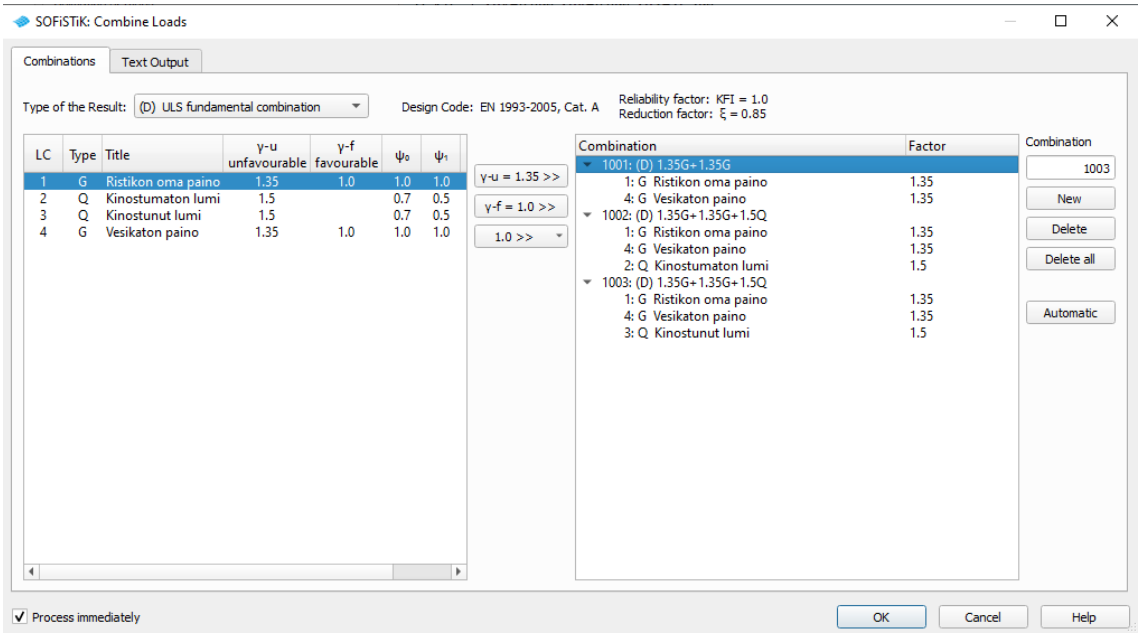
Kuva 17. SOFiSTiK Strctural Desktop 2020 ohjelmistossa esimerkkitapauksen ristikkorakenne.

2.3 Kuormitus yhdistelmät

Luodaan taulukon 3 mukaiset kuormitusyhdistelmät SOFiSTiKiin manuaalisesti (kuva 18). Grasshopperiin määriteltiin kuormituksille tyypit ”G” ja ”Q”, jotka viittaavat pysyvään- sekä muuttuvaan kuormitukseen. Tällöin SOFiSTiK tunnistaa eri tyypit, ja osaa antaa varmuuskertoimet Eurokoodin mukaan. SOFiSTiKissa on paljon eri esiohjelmoituja kuormitusyhdistelmiä, joita voidaan tarkastella useissa eri rajatilamitoituksissa.

Taulukko 3. Kuormitusyhdistelmät.

Kuormitus yhdistelmän numero	**Yhdistely**	**Sisältävät kuormitukset**
1001	1.35G+1.35G	Ristikon omapaino, vesikaton paino
1002	1.35G+1.35G+1.5Q	Ristikon omapaino, vesikaton paino, kinostumaton lumi
1003	1.35G+1.35G+1.5Q	Ristikon omapaino, vesikaton paino, kinostunut lumi

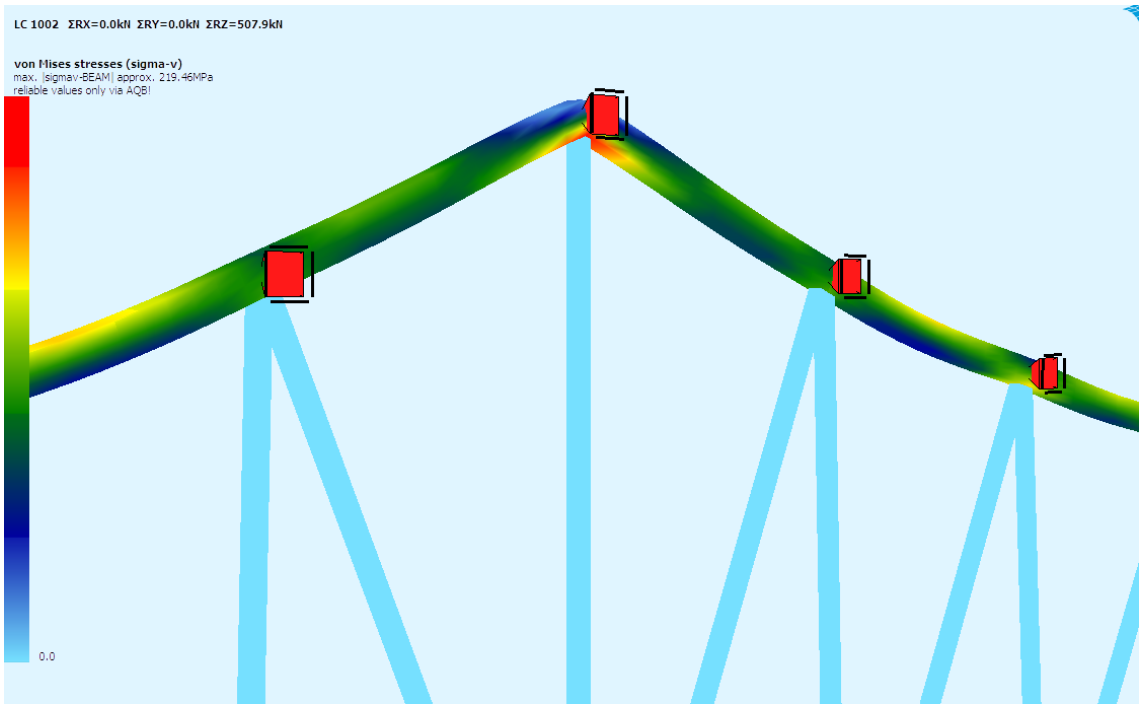


Kuva 18. Kuormitusyhdistelmät.

2.4 Laskennan tulokset

Rakenneanalyysin tuloksia on helppo tarkastella SSD:ssä. Ohjelmisto havainnollistaa hyvin ristikon siirtymät, ja näyttää värikoodilla suhteellisia jännityksiä ristikon eri osissa. Tarkemmin ristikon osiin kohdistuvia voimia ja solmusiirtymiä voidaan tarkastella *Grahphic* ikkunassa. *Graphic* ikkunan sivussa olevasta valikosta voidaan valita kuormitustapaus, ja sen mukaiset voimat, momentit ja solmusiirtymät. Esitystapa on selkeä, ja tulokset ovat helppo ottaa raporttiin.

SOFiSTiKista on suoraan nähtävissä vain maksimi von Mises -jännitys. Tarkempi yksittäisissä rakenneosissa vallitseva jännitys jätetäänkin tämän tutkimuksen ulkopuolelle. SOFiSTiK varoittaakin, että ilmoitettu maksimijännitys on arvio, ja että tarkemmat jännitykset saadaan AQB:n kautta. AQB jätetään kokonaan tämän työn ulkopuolelle. Kuvasta 19 nähdään, että von Mises -maksimijännitys syntyy yläpaarteiden liitoksen alapuolelle, joka on vaarallisimmassa kuormitustapauksessa suuruudeltaan 219.46 MPa.



Kuva 19. Von Mises -maksimijännitys ristikon harjalla, yläpaarteen alapinnalla.

2.5 Optimointi

Rakennetta olisi helppo lähteä nyt optimoimaan muuttamalla Grasshopperissa parametreja, sillä uusi malli päivittyy SSD:hen aina, kun *SOFiSTiK Project*-komponentin painiketta *Calculate* painetaan. Rakennetta voitaisiin lähteä optimoimaan esimerkiksi muuttamalla ristikon tiheyttä, mallia, poikkileikkauksen profiilia tai materiaalia. Myös esimerkiksi ristikon diagonaalien ja vertikaalien kiinnityksen vapausasteiden vaikutuksen tarkastelu on helppoa, eli käyttäytyykö ristikko ideaalisesti vai syntyykö diagonaaleihin ja vertikaaleihin myös momenttia.

3 JOHTOPÄÄTÖKSET

Grasshopper osoittautui helposti opittavaksi visuaalisen ohjelmoinnin työkaluksi. Toisaalta on hyvä huomioida, että minulla oli *ohjelmoinnin perusteet* -kurssi käytynä, jossa opeteltiin Python ohjelmointikielen perusteita. Lisäksi olin tutustunut Autodeskin Dynamo -ohjelmistoon jonkin verran ennen Grasshopperin käytön opettelua. Edellä mainitut asiat edesauttoivat varmasti Grasshopperin käytön oppimisessa. Ristikon geometrian määrittely Grasshopperilla oli suhteellisen helppoa ja nopeaa, ja erilaisiin ongelmatilanteisiin löytyi yleensä netistä vastaus nopeasti.

SOFiSTiKin Grasshopper-komponenteissa on kuitenkin vielä kehittämisen varaa. Esimerkiksi *SOFiSTiK Project*-komponenttiin tulee vetää johdot siinä järjestyksessä, kun laskenta halutaan suoritettavan SOFiSTiKissa. Tämä tarkoittaa sitä, että koodista ei suoraan nähdä mitä laskennassa tapahtuu, vaan se täytyy kokeilla, tai vetää johdot uudelleen, jotta voidaan olla varmoja laskentajärjestyksestä. Grasshopperiin ei myöskään tullut lisäosan mukana kaikkia tarvittavia komponentteja, joilla laskentamalli olisi voitu määritellä täysin Grasshopperin puolella. Nyt esimerkiksi kuormitustapaukset ovat helpompi määritellä SSD:ssä, kuin Grasshopperissa.

Valmis Grasshopper-koodi on helppo ja nopea tapa mallintaa kattoristikko, erityisesti silloin, kun on itse tehnyt koodin ja tuntee sen hyvin. Tässä työssä luotua koodia ei ole arvioitu ulkopuolisen toimesta, mutta koodin tekijänä koin sen selkeäksi.

SOFiSTiKin puolella ilmeni joitakin ongelmia rakenneanalyysissä. Esimerkiksi jos alapaarre oli täysin suora, syntyi siitä epästabiilisuus alapaarteeseen. Ongelma voitiin välttää nostamalla alapaarteen harjakorkeutta (ahk) 0.001 m ylöspäin. Ongelman syyhyn ei löytynyt ratkaisua, eikä sen selvittämiseen laajemmin pyrittykään. Osa ilmenneistä ongelmista johtui kokemukseni puutteesta SSD:n käytössä. Myös CADiNP-ohjelmointikieleen kannattaa perehtyä, mikäli SOFiSTiKia aikoo käyttää.

Kandidaatin työn aiheeksi Grasshopperin ja SOFiSTiKin käytön tutkiminen osoittautui liian laajaksi. Tutkimuksen aikana jouduin ohittamaan monia ilmenneitä ongelmia, jotta työmäärä ei kasvaisi liian suureksi. Molemmissa tutkituissa ohjelmistoissa oli paljon ominaisuuksia joihin perehtyminen olisi vaatinut paljon aikaa. Tutkimuksen

lopputulokseksi saatiin kuitenkin toimiva Grasshopper-koodi, jolla on helppo mallintaa erilaisia ristikoita, ja siirtää malli sieltä SOFiSTiKiin rakenneanalyysia varten.

4 YHTEENVETO

Tutkimuksessa tutustuttiin parametriseen rakennesuunnitteluun opinnäytetöiden pohjalta. Tarkoituksena oli selvittää, mitä parametrinen rakennesuunnittelu käytännössä on, ja mitkä ovat menetelmän heikkoudet ja vahvuudet, sekä millaisissa projekteissa parametrinen suunnittelu on koettu hyödylliseksi.

Useissa käytetyissä lähteissä todettiin parametrisella rakennesuunnittelulla saavutettavan merkittäviä hyötyjä. Hyviksi puoliksi koettiin esimerkiksi geometrisesti haastavien rakenteiden nopeampi mallinnus verrattuna perinteiseen mallinnusmenetelmään, nopeampi muutosten toteutus, usean vaihtoehtoisen ratkaisun vertailu ja samankaltaisten rakenteiden suunnittelun nopeutuminen.

Parametrisen suunnittelun heikkouksiksi todettiin mm. aiheeseen liittyvä osaaminen. Kyseessä on suhteellisen uusi suunnittelumenetelmä, jota ei opeteta vielä laajassa mittakaavassa. Suunnittelijoiden osaaminen on siis pääasiassa itseopiskelun varassa. Lisäksi parametrisen suunnittelun käytön valintaan projektissa liittyy riskejä, sillä vielä on varsin vähän kokemusperäistä tietoa kuluvista työmääristä kyseistä menetelmää käytettäessä.

Tutkimuksessa luotiin parametrinen laskentapohja kattoristikon mallinnukseen ja analyysiin. Geometria, profiilit, materiaalit ja kuormitukset luotiin Grasshopperissa, joka on visuaalinen ohjelmointityökalu. Rakenneanalyysi suoritettiin SOFiSTiKissa, johon Grasshoppermalli linkitettiin.

Työssä selostettiin suurpiirteisesti koodin tärkeimmät kohdat ja niiden toimintaperiaatteet. Yksittäisten komponenttien toimintaan ei keskitytty tässä työssä, vaan koodi luotiin valmiiden Grasshopper-komponenttien sekä SOFiSTiKin asennuksen mukana tulleiden komponenttien avulla. Tämän työn avulla Grasshopperia osaavan käyttäjän pitäisi pystyä luomaan parametrinen kattoristikko.

Grasshopper osoittautui työssä helpoksi ja nopeaksi ohjelmistoksi mallintaa ristikkorakenne. Koin sen käytön sujuvaksi, eikä ohjelmisto kaatunut kertaakaan työn

aikana. SOFiSTiKin käyttö Grasshopperin avulla ei ollut yhtä helppoa ja varmaa, mutta työssä saatiin toteutettua toimiva ristikon suunnitteluohjelma.

LÄHDELUETTELO

Lalla A., 2017. Kantavien rakenteiden parametrinen suunnittelu ja mallintaminen. Diplomityö, Tampereen teknillinen yliopisto, rakennustekniikan osasto, Tampere. 98 + 6 s.

SFS-EN 1991-1-3 + AC + A1, 2015. Eurokoodi 1: Rakenteiden kuormat. Osa 1–3: Yleiset kuormat. Lumikuormat, 2.painos. Helsinki: Suomen Standardoimisliitto SFS, 47 + 34 s.

Tanska T. & Österlund T., 2014. Algoritmit puurakenteissa: menetelmät, mahdollisuudet ja tuotanto, B 32, 1. painos DigiWoodLab, Oulun yliopisto, Arkkitehtuurin tiedekunta, Oulu. 176 s.

Toola, M., 2017. Teräsrakenteisten siltojen parametrinen mallinnus. Opinnäytetyö, Tampereen ammattikorkeakoulu, Rakennustekniikan osasto, Talonrakennustekniikka. 34 s.

Vähänen, P., 2019. Parametrisen suunnittelun hyödyntäminen teräsbetonisten runkorakenteiden luonnossuunnittelussa. Diplomityö, Oulun yliopisto, Rakennus- ja yhdyskuntatekniikan osasto. Oulu. 84 s.

Wikipedia, 2020. Algoritmi. Saatavissa: https://fi.wikipedia.org/wiki/Algoritmi [viitattu 22.3.2021].

WSP, 2021. Kruununvuorensilta. Saatavissa: https://www.wsp.com/fi-FI/projects/kruunuvuorensilta [Viitattu 3.5.2021].

Ympäristöministeriö, 2019. Suomen rakentamismääräyskokoelma. Rakenteiden lujuus ja vakaus. Rakenteiden kuormat. Helsinki. 51 s.